MOSAICING OF AERIAL IMAGES USING FEATURE POINT EXTRACTION FOR USE IN VARIATION DETECTION

Syndicate Members
Syed Akbar Mehdi
Waqar Ali
Chaudhry Nauman Zafar
Muhammad Sohail

A Thesis in Partial fulfillment of the requirements for the degree of
B.E. (Computer Software),
National University of Sciences and Technology (NUST),
APRIL 2005

ABSTRACT

Our aim in this project is specifically to design a system that will automatically mosaic aerial images. These images may be from a satellite or any other source such as an Unmanned Aerial Vehicle (UAV). Any such vehicle when sent into air to photograph a certain area takes several pictures as it flies above. For proper analysis these images need to be mosaiced in order to create a single bigger picture of the whole area.

The images have areas which overlap or are present in more than one image. Before all the images are combined these overlaps need to be automatically detected the images be joined in such a way that the image contains no doubling. The number of images may be quite large and the matching process may take quite some time. Efficient algorithms need to be designed and implemented in order to piece together all the pictures correctly in a short time without much error.

In order to resolve the above difficulties, the techniques of Feature Point Extraction are used. This relies on accurate detection of image features such as detecting corners. After the feature points have been extracted we can use the analyzed data to help us mosaic the images.

The resulting mosaic can be used in a number of ways. Suppose that the same area is photographed once again. The new pictures can again be analyzed in the same way to form another mosaic. The two mosaics can then be compared using mathematical techniques to detect any changes in the area and report to the user. Some examples of these changes can be construction of a building or the digging of a bunker etc.

DEDICATION

In the name of Allah, the most gracious and the most merciful

We are highly indebted to Lt.Col.Naveed Sarfraz Khattak, our mentor, without whose unwavering and continued guidance, we would have not been able to complete this project.

To our parents, without whose unflinching support and unstinting cooperation, a work of this magnitude would not have been possible.

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institution or elsewhere

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

To design a system for the automated feature point extraction and mosaicing of aerial images and use the results for the purpose of change detection in various versions of the same mosaic.

1.2 PURPOSE

Our aim in this project is specifically to design a system that will automatically mosaic aerial images. These images may be from a satellite or any other source such as an Unmanned Aerial Vehicle (UAV). Any such vehicle when sent in to photograph a certain area takes several pictures as it flies above. For proper analysis these images need to be mosaiced in order to create a single bigger picture of the whole area. The design of such a system has inherent difficulties because of two reasons.

Firstly, the images have areas which overlap or are present in more than one image. Before all the images are combined these overlaps need to be automatically detected the images be joined in such a way that the image contains no doubling.

Secondly, the number of images may be quite large and the matching process may take quite some time. Efficient algorithms need to be designed and implemented in order to piece together all the pictures correctly in a short time without much error.

In order to resolve the above difficulties, the technique of Feature Point Extraction was used. This relies on accurate detection of image features such as detecting corners. After the feature points have been extracted so that one can use the analyzed data to help us mosaic the images.

The resulting mosaic can be used in a number of ways. Suppose that the same area is photographed once again. The new pictures can again be analyzed in the same way to form another mosaic. The two mosaics can then be compared using mathematical techniques to detect any changes in the area and report to the user. Some examples of these changes can be construction of a building or the digging of a bunker etc.

1.3 OBJECTIVES

Specifically the project has the following objectives:

1.3.1 GEOMETRIC CORRECTIONS

The first step is to correct geometric deformations using image data and/or camera models. This is necessary because images taken from Unmanned Aerial Vehicles (UAVs) are from various positions. Depending on the elevation and azimuth of the aircraft and camera system, these images may exhibit a variety of perspective distortions, and the orientation, scale and position of the images are not available. Hence, to register such images, one has to determine the affine transformation between the images (scale, position and rotation).

1.3.2 IMAGE MOSAICING

The second step is the mosaicing of the images. For this purpose the feature points have to be extracted in order to correctly match and mosaic them. Hence this part has two steps. The first step is the feature point extraction and the second is mosaicing based on the feature points.

Our core objective will be the achievement of the second task i.e. image mosaicing. The assumption will be that the first task has been completed. Also the third task is included as a further advancement that can be completed once the second has been successfully achieved.

1.3.3 CHANGE DETECTION

The last step in our project, and of course, the one with a wide number of military and civil applications is the change detection. By change detection, one means finding the difference, if any, between two mosaiced images of the same area, photographed at different points in time.

1.4 IMPLEMENTATION

The system has been implemented using digital image processing techniques. The testing of the mathematical techniques was done on MATLAB while the final system was designed in using Visual C++ using the Microsoft Vision SDK library as a base.

1.5 OVERVIEW OF IMAGE MOSAICING

Registration and mosaicing of images have been in practice since long before the age of digital computers. The limited flying heights of the early airplanes and the need for large photo-maps, forced imaging experts to construct mosaic images from overlapping photographs. This was initially done by manually mosaicing images which were acquired by calibrated equipment. The need for mosaicing continued to increase later in history as satellites started sending pictures back to earth. Improvements in computer technology became a natural motivation to develop computational techniques and to solve related problems.

Image Mosaicing has important applications in both military and non-military domains. The construction of mosaic images and the use of such images on several computer vision/graphics applications have been active areas of research in recent years. Image-based rendering has become a major focus of attention combining two complementary fields: computer vision and computer graphics. In computer graphics applications images of the real world have been traditionally used as environment maps. These images are used as static background of synthetic scenes and mapped as shadows onto synthetic objects for a realistic look with computations which are much more efficient than ray tracing.

Among other major applications of image mosaicing in computer vision are image stabilization, resolution enhancement, video processing (e.g. video compression, video indexing).

The images shown on the next page demonstrate the actual working of image mosaicing. Here, both the pictures have some area in common which has been detected by our program and a mosaic has thus been created devoid of any overlapping area.

If the area of interest is simply too large to be covered by a single photo, several adjoining photos can be combined to form a "mosaic" image. A mosaic of this type is defined in the dictionary as "a composite map made of aerial photographs". If the imagery has been digitally processed and manipulated by computer software, it can be made to look like a single photograph with no apparent seam lines where several photos were "stitched" together.

The mosaiced image can also be "rectified" using one of several methods to remove varying amounts of distortion and displacement caused by variables such as the tip and tilt of the aircraft, changes in elevation on the ground, and imperfections in camera lenses. Prior to the development of computer assisted image processing, mosaics were made by carefully cutting or tearing paper photographs along their edges in an irregular fashion, and gluing them down in such a way that the cut or tear lines were overlapped on adjacent photos and blended-in to hide the lines as much as possible.

Image mosaicing is an active area of research in computer vision. The various methods adopted for image mosaicing can be broadly classified into direct methods and feature based methods. Direct methods are found to be useful for mosaicing large overlapping regions, small translations and rotations. Feature based methods can usually handle small overlapping regions and in general tend to be more accurate but computationally intensive. Some of the basic problems in image mosaicing are alignment, adjustment, automatic selection of images to be blended and exposure compensation.

Even after good global alignment, some pixel might not align in the two images. This might cause ghosting or blur in the blended image. Automatic selection of images to blend from a given set of images is also another area to be taken care of. After one of the images has been transformed using the homography

calculated above a decision needs to be made about the color to be assigned to the overlapping regions. Blending also becomes important when there exists a moving object in the images taken.

Most cameras have an automatic exposure control. The images taken can therefore be of variable brightness in the overlapping region which might cause the mosaic to look unrealistic.

1.6 SYSTEM OVERVIEW

The system is fed with raw images. The images may be disoriented, unaligned or may not be parallel. The first step, therefore, is to perform geometric corrections on the images. Geometric corrections part of the project is being carried out by another research group. It is, therefore, assumed that the images fed to our system are free from any geometric disturbances.

Once the images have been fed, the next step is to extract features from the image set. By features, it is believed that a sudden change in the intensity of pixel values occurs. There are two types of features that may be detected in an image. These may be either corners or edges. This is the first and foremost task accomplished by our software.

After the detection and identification of corners, the next aim is to perform feature point matching on the image set. The feature points are matched in the two images by calculating differentials at the pixel locations. If the difference is less than a specified threshold, then it means that the features have been matched successfully. Another confirmatory check is applied in the form of correlation and ten nearest neighbours in both the images are checked to see if they are similar.

The most important phase of our project is the image mosaicing part. Once two images have been identified to be containing similar feature points, and then the overlapping area is calculated. A new image is created and the two images, minus the overlapping area, are copied into that image. The same process is repeated for all the images in the image set and each time, two images are mosaiced into one.

The last step is the variation detection. Two mosaics of the same area are checked to determine variation, if any, from the previous images of the same area. For this purpose, differentials of each pixel position in the images are calculated and the values are stored.

Where the difference of the same pixel position differential is greater than a specified threshold, a change is detected. The change is shown by the marking of red blocks on the image area where there is some change.

Both, the mosaiced images and the variation detected images can be saved to permanent storage, if desired by the user.

The flow diagram is as shown in figure 1.1.

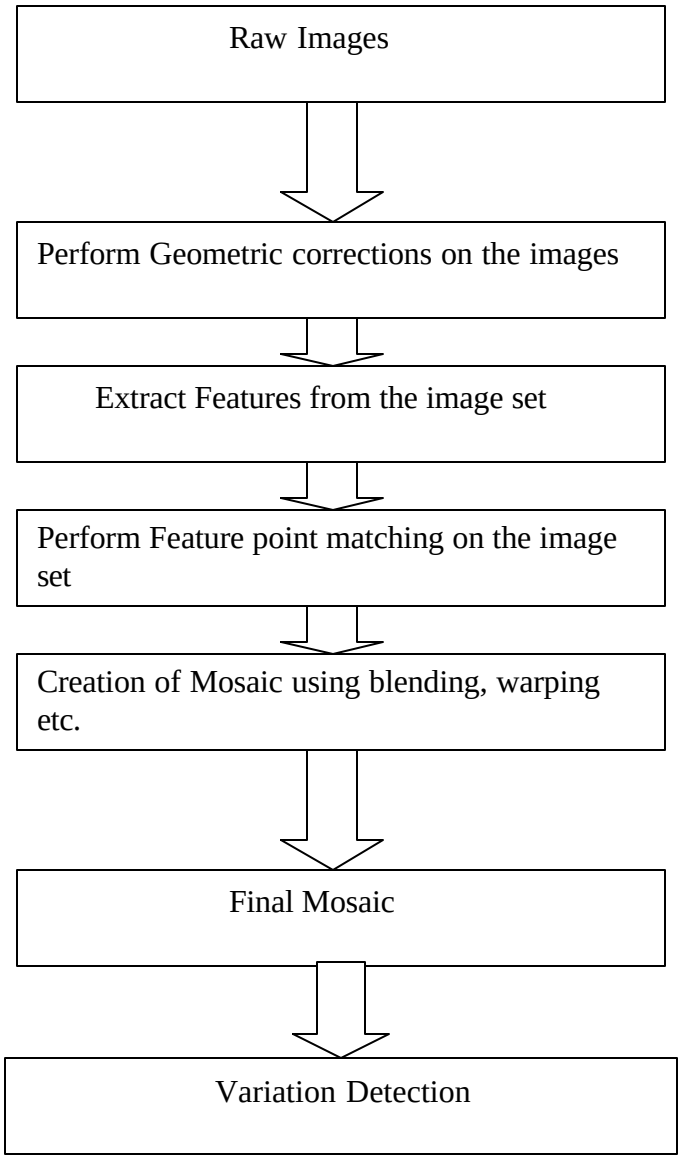

Figure 1.1 System Overview

CHAPTER 2

LITERATURE REVIEW

2.1 FEATURE EXTRACTION

2.1.1 REQUIREMENTS

Corner detection should satisfy a number of important criteria. This may include conditions such as the true corners should be detected, no false corners should be detected, corner points should be well localized, corner detector should be robust with respect to noise and that the corner detector should be efficient.

2.1.2 THE PLESSEY FEATURE POINT DETECTOR

Harris and Stephens described what has become known as the Plessey feature point detector. The outline of how it works can be best understood if the following matrix is considered

$$\mathrm{M} = \begin{bmatrix} \left(\frac{\partial I}{\partial x}\right)^2 & \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) \\ \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) & \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix} \qquad (1)$$

where I(x; y) is the grey level intensity. If at a certain point the two Eigen values of the matrix M are large, then a small motion in any direction will cause an important change of grey level. This indicates that the point is a corner. The corner response function is given by:

$$R = \det \mathrm{M} - k(\mathrm{trace}\mathrm{M})^2 \qquad (2)$$

where k is a parameter set to 0.04 (a suggestion of Harris). Corners are defined as local maxima of the corner-ness function. Sub-pixel precision is achieved through a quadratic approximation of the neighborhood of the local maxima. To avoid corners due to image noise, it can be interesting to smooth the images with a Gaussian filter. This should however not be done on the input images, but on images containing the squared image derivatives .In practice often far too much corners are extracted. In this case it is often interesting to first restrict the numbers of corners before trying to match them. One possibility consists of only selecting the corners with a value R above a certain threshold. This threshold can be tuned to yield the desired number of features. Since for some scenes most of the strongest corners are located in the same area, it can be interesting to refine this scheme further to ensure that in every part of the image a sufficient number of corners are found.

2.1.3 THE SUSAN DETECTOR

SUSAN (**Smallest Univalued Segment Assimilating Nucleus**) presents us with an entirely different approach to low level image processing compared to all pre- existing algorithms. It provides corner detection as well as edge detection and is more resistant to image noise although no noise reduction (filtering) is needed. The concept of each image point having associated with it a local area of similar brightness is the basis for the SUSAN principle. If the brightness of each pixel within a mask is compared with the brightness of that mask's nucleus then an area of the mask can be defined which has the same (or similar) brightness as the nucleus. This area of the mask shall be known as the USAN", an acronym standing for **Univalued Segment Assimilating Nucleus**".

Computing USAN for every pixel in the digital image provides us with a way to determine the edges inside it. The value of USAN gets smaller on both sides of an edge and becomes even smaller on each side of a corner. Hence one is looking for the Smallest USAN (or SUSAN for short). The local minima of the USAN map represent corners in the image. The reason that this method stays resistant to noise is the lack of computing spatial derivatives of the image intensity.

2.1.4 THE CURVATURE SCALE SPACE (CSS) CORNER DETECTOR

The curvature scale space technique is suitable for recovering invariant geometric features (curvature zero-crossing points and/or extrema) of a planar curve at multiple scales. The CSS corner detector works in a sequence.

First of all, the rule is to extract the edge contours from the input image using any good edge detector such as Canny is made. Then, small gaps in edge contours are filled. When the gap forms a T-junction, it as marked as a T-corner. Curvature on the edge contours at a high scale is computed. The corner points are defined as the maxima of absolute curvature that are above a threshold value. The corners are tracked through multiple lower scales to improve localization. T-corners are compared to the corners found using the CSS procedure and remove very close corners.

Experimental results show this algorithm spends most of its time (80%) detecting the edges in the image. Faster edge detectors may be used. The local maxima of absolute curvature are the possible candidates for corner points. A local maximum is either a corner, the top value of a rounded corner or a peak due to noise. The latter two should not be detected as corners. The curvature of a real corner point has a higher value than that of a rounded corner or noise. However, as shown in figure 2.1, sometimes, a corner is detected twice.

The corner points are also compared to the two neighboring local minima. The curvature of a corner should be twice that of one of the neighboring local minima. This is because when the shape of the contour is very round, contour curvature can be above the threshold.

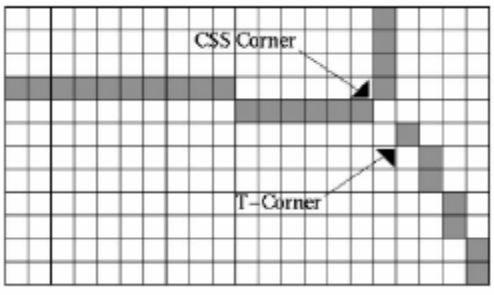

Figure 2.1 A corner is detected twice

2.1.5 IPAN 99

The acronym for this method stands for **Image and Pattern Analysis** group and was developed in 1999 at The Hungary Academy of Science. It is fast and efficient algorithm for detection of high curvature points. The curve has to be generated previously using an edge detector. It is not required to be a closed curve. In the first pass the sequence of points is scanned and candidate corner points are selected. In each curve point p the detector tries to inscribe in the curve a variable triangle (p_ ,p ,p+). Because the points are kept with their Cartesian coordinates, the angle be easily computed.

Triangles are selected starting from point p outward and stop on the conditions mentioned above. In that way a number of admissible triangles are defined. At a neighborhood of points, only one of these admissible triangles is selected – the one which has the smallest value for the angle. A value of sharpness is assigned to p.

In the second pass the selection is refined and points that give the strongest response are marked as corners in the curve. This is done by selecting only points which have sharpness greater than that of their neighbors. Figure 2.2 shows the working of IPAN 99.

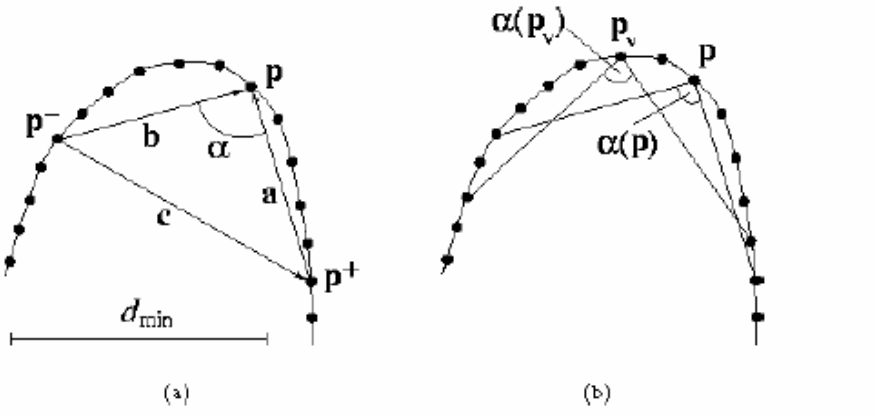

Figure 2.2 Working of the IPAN 99

2.2 IMAGE MATCHING

In photogrammetric and remote sensing, matching can be defined as the establishment of the correspondence between various data sets. The matching problem is also referred to as the correspondence problem. The data sets can represent images, but

also maps, or object models and GIS data. Many steps of the photogrammetric processing chain are linked to matching in one way or another. Examples include the reconstruction of the interior orientation: the image of a fiducial is matched with a two-dimensional model of the fiducial; relative orientation and point transfer in aerial triangulation: parts of one image are matched with parts of other images in order to generate tie points; absolute orientation: parts of the image are matched with a description of control features, mostly ground control points; generation of **digital terrain models (DTM):** parts of an image are matched with parts of another image in order to generate three-dimensional object points; and finally the interpretation step: parts of the image are matched with object models in order to identify and localize the depicted scene objects.

Looking at this large variety of tasks it comes as no surprise that matching has long been and still is one of the most challenging tasks in photogrammetric research and development. In this paper an overview is given of a more specific class of matching algorithms usually called *digital image matching*. Digital image matching *automatically establishes the correspondence between primitives extracted from two or more digital images depicting at least partly the same scene*. The primitives can be gray level windows or features extracted from the images. Thus, all input data sets are images or parts thereof. Objects as such need not be modeled explicitly. It should be kept in mind, however, that each algorithm uses at least an implicit model of the object surface, since it is the object surface which is depicted in the images.

In photogrammetric and remote sensing, image matching is employed for relative orientation, point transfer in aerial triangulation, scene registration and DTM generation. Also, the reconstruction of the interior orientation falls within the category of image matching, since the model of a fiducial is usually represented as a gray value image.

2.2.1 IMAGE MATCHING HISTORY

First solutions for image matching have been suggested already in the late fifties (Hobrough 1959, he still used analogue images and procedures). Since then a steady increase in the interest for image matching has occurred, and the question may be asked, why image matching has not been solved long ago. A first answer can be given by considering the information content of the most elementary primitive in the input data set, namely a pixel. An aerial image scanned with 15 um contains approximately 235.000.000

pixels, and each gray value usually lies in the range of 0 to 255. Assuming an equal distribution of the gray values the image contains roughly 920.000 pixels of each gray value. This little computation demonstrates that matching on the basis of single pixels is certainly impossible. It also exemplifies two fundamental problems of image matching.

First problem is that ambiguous solutions may occur, if image matching is tackled using local information, and the second one is that computational costs are high and have to be controlled.

A more realistic approach is that of cross correlation. In order to compute the cross correlation function of two windows, a template window is shifted pixel by pixel across a larger search window and in each position the cross correlation coefficient ñ between the template window and the corresponding part of the search window is computed according to equation 2.1. The maximum of the resulting cross correlation function defines the position of the best match between the template and the search window.

$$\rho = \frac{\sum_{r=1}^{R}\sum_{c=1}^{C}(g_1(r,c)-\mu_1)(g_2(r,c)-\mu_2)}{\sqrt{\sum_{r=1}^{R}\sum_{c=1}^{C}(g_1(r,c)-\mu_1)^2 \sum_{r=1}^{R}\sum_{c=1}^{C}(g_2(r,c)-\mu_2)^2}} \quad ; \quad -1 \le \rho \le 1$$

Equation 2.1

$g_1(r,c)$	individual gray values of template matrix
u_1	average gray value of template matrix
$g_2(r,c)$	individual gray values of corresponding part of search matrix
u2	average gray value of corresponding part of search matrix
R, C	number of rows and columns of template matrix

The principle of cross-relation function is shown in figure 2.3 while cross-relation functions are shown in figure 2.4.

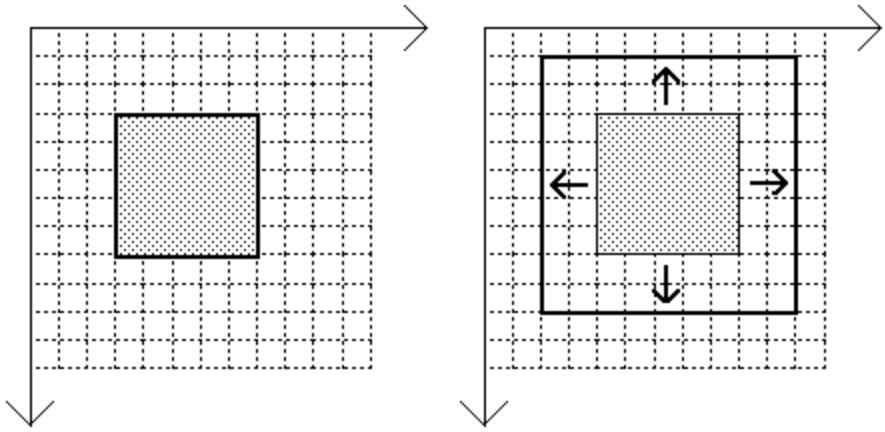

Figure 2.3: Principle of cross correlation

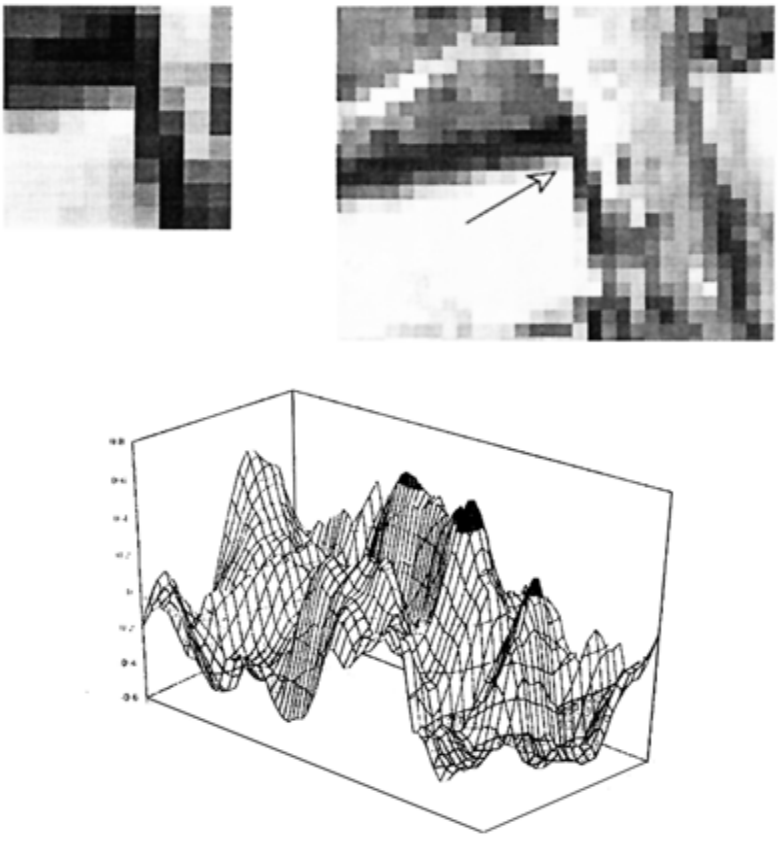

Figure 2.4: a (top left) left image; b (top right) right image; c (bottom) cross correlation function

A typical result of cross correlation is shown in figure 2.4. Figure 24a shows a typical small template window of the left image of an aerial stereo pair, figure 24b

depicts the corresponding larger search window in the right image. In figure 2.4c a plot of the cross correlation function of the two windows is shown. The cross correlation coefficient is a simple but widely used measure for the similarity of different image windows As can be seen in figure 2.4c, the spatial variation of the cross correlation coefficient can be extensive making it a difficult task to find its maximum. During this projection information is lost. This is most evident in the case of occlusions. Image matching belongs to the class of so called inverse problems, which are known to be ill-posed. A problem is ill-posed, if no guarantee can be given that a solution exists, is unique, and is stable with respect to small variation in the input data. Image matching is ill-posed, because for a given point in one image, a corresponding point may not exist due to occlusion, there may be more than one possible match due to repetitive patterns or a semi-transparent object surface, and the solution may be unstable with respect to noise due to poor texture.

In order to find the solution of an ill-posed problem one usually has to deal with an optimization function exhibiting many local extrema (as can be seen in figure 2.4c), and thus a small pull-in range. Therefore, stringent requirements may exist for initial values for unknown parameters to be determined. Moreover, usually there is a large search space for these parameters, and numerical instabilities may arise during the computations.

Ill-posed problems can be converted to well-posed problems by introducing additional knowledge about the problem. Fortunately, a whole range of assumptions usually holds true when dealing with photogrammetric imagery.

The assumptions usually made are that the gray values of the various images have been acquired using one and the same or at least similar spectral band(s), the illumination together with possible atmospheric effects are constant throughout the time interval for image acquisition, the scene depicted in the images is rigid, i.e. it is not deformable; this implies that objects in the scene are rigid, too, and do not move, the object surface is piecewise smooth , the object surface is opaque, the object surface exhibits a more or less diffuse reflection function and that initial values such as the approximate overlap between the images or an average object height are known

Depending on the actual problem at hand additional assumptions may be introduced, and some points of the list may be violated. It is this mixture of necessary assumptions

which makes the design of a good image matching algorithm difficult, and has lead to the development of different algorithms in the past.

2.2.2 ANALYSIS OF IMAGE MATCHING ALGORITHMS

Most matching algorithms proposed in the literature implicitly or explicitly contain a combination of assumptions about the depicted scene and the image acquisition. Rather than trying to describe these algorithms as a whole it seems more appropriate to decompose them into smaller modules and discuss those.

The factors that have to be answered are that to be answered are that which primitives are selected for matching, which models are used for defining the geometric and radiometric mapping between the primitives of the various images, how is the similarity between primitives from different images measured, and how is the optimal match computed and which strategy is employed in order to control the matching algorithm.

Based on these points, an analysis of different algorithms is presented.

2.2.2.1 DIFFERENT MATCHING PRIMITIVES

The distinction between different matching primitives is probably the most prominent difference between the various matching algorithms. One of the reasons is that this selection influences in part the answers to the other questions. The primitives fall into two broad categories: either windows composed of gray values or features extracted in each image a priori are used in the actual matching step. The resulting algorithms are usually called ***area based matching (ABM)***, and ***feature based matching (FBM),*** respectively. Note that when talking about ABM or FBM not only the selection of the primitives, but the whole matching process is referred to.

In both cases there is a choice between local and global support for the primitives. The terms local and global are not sharply defined. Local refers to an area seldom larger than about 50 * 50 pixels in image space, global means a larger area and can comprise the whole image.

2.2.2.1.1 Gray Value Windows as Primitives

Small windows composed of gray values serve as matching primitives. The window centre, possibly weighted e.g. with respect to the gray value gradient can be used for the definition of the location of a point to be matched. The gray values are regarded as

quantized samples of the continuous brightness function in image space, and concepts of signal processing can be employed for further computations.

The windows can be extracted very fast, and the actual matching methods are rather straightforward. Also, ABM has a high accuracy potential in well-textured image regions, and in some cases the resulting accuracy can be quantified in terms of metric units. Disadvantages of ABM are the sensitivity of the gray values to changes in radiometry e.g. due to illumination changes, the large search space for matching including various local extrema, and the large data volume which must be handled. Blunders can occur in areas of occlusions, and poor or repetitive texture.

ABM is usually based on local windows. One example is cross correlation, another one is the original least squares matching approach. ABM can also be carried out globally using connected windows. In this case poor and repetitive texture can be successfully dealt with to a certain extend.

2.2.2.1.2 Features as Primitives

In FBM features are extracted in each image individually prior to matching them. Local features are points, edge elements, short edges or lines, and small regions. Global features comprise polygons and more complex descriptions of the image content called structures. Features should be distinct with respect to their neighborhood, invariant with respect to geometric and radiometric influences, stable with respect to noise, and seldom with respect to other features.

Each feature is characterized by a set of attributes. The position in terms of its image coordinates is always present. Further examples for attributes are the edge orientation and strength (gradient across the edge) for edge elements, the length and curvature of edges and lines, the size and the average brightness for regions.

Global features are usually composed of different local features. Besides the attributes of the local features, relations between these local features are introduced to characterize global features. These relations can be geometric such as the angle between two adjacent polygon sides or the minimum distance between two edges, radiometric such as the difference in gray value or gray value variance between two adjacent regions or

topologic, such as the notion that one feature is contained in another. Matching with global features is also referred to as ***relational matching***.

The result of feature extraction is a list containing the features and their descriptions for each image. Only these lists are processed further. It should be noted that the features are discrete functions of position: after feature extraction a feature either exists at a given position or it does not.

Features are more abstract descriptions of the image content. As compared to gray value windows features are in general more invariant with respect to geometric and radiometric influences. Feature extraction schemes are often computationally expensive and require a number of free parameters and thresholds which must be chosen a priori. In some cases a shift in the feature position is introduced during the extraction. If this shift is corrected for local features have a high accuracy potential. It is, however, difficult to quantify this accuracy in metric units. In areas of low texture the density of extracted features is usually sparse. For local features, seldom ness is difficult to achieve, and a large data volume must be handled. Global features are more seldom and thus provide a better basis for a reliable matching. However, it is difficult to define and extract global features, and they tend to be more application dependent then local features.

Local features have been used for matching e.g. by Barnard, Thompson (1980); Förstner (1986) and Hannah (1989). In each case points were selected as features. Vosselman (1992); Vosselman, Haala (1992); Cho (1995) and Wang (1995) dealt with relational matching involving global features. Schenk et al. (1991) used a combination of global and local features.

2.2.2.2 MODELS FOR THE MAPPING OF PRIMITIVES

The mapping between the primitives of the various images is defined via two models: a sensor model, and a model for the object surface. Simple two-dimensional transformations from one image to the next such as a two-dimensional translation or an affine transformation implicitly contain a combination of these two models. They are rough approximations of the situation during image acquisition and should only be used, if the selected matching primitives have local support. If, on the other hand, primitives with global support are used, the mapping between the images must be modeled more

rigorously. Usually, the sensor and the object surface model are specified separately. As shown in figure 2.5, any two-dimensional transformation can be constructed from a sequence of two three-dimensional transformations: one from image space into object space, and a second one into the image space of the other image.

There is an advantage if the mapping between the primitives - local or global - is formulated in terms of object space parameters which are common for more than two images: multiple images can be matched simultaneously. This results in a higher redundancy for the matching problem and thus a greater reliability is achieved for the results. Multi image matching using ABM has been shown to be superior to matching of two images.

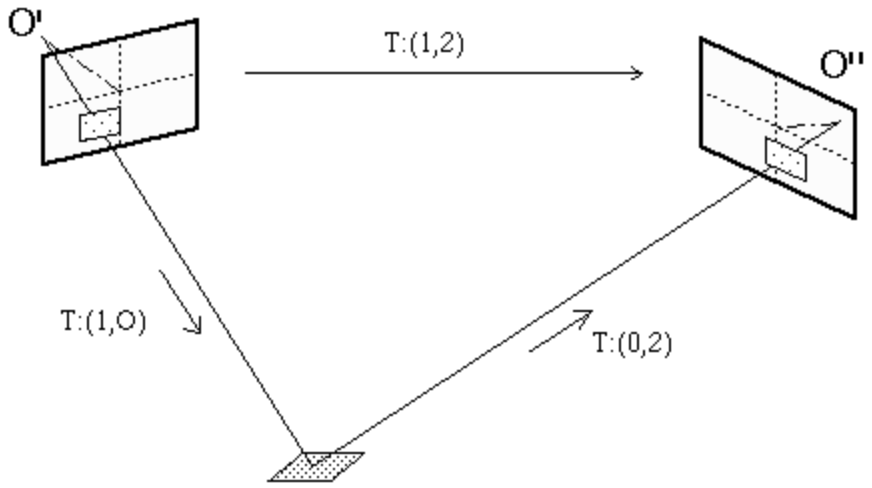

Figure 2.5: A 2D transformation can be expressed as a combination of two 3D transformations

2.2.2.2.1 Sensor model and epipolar constraint

In many cases a central perspective projection can be assumed when dealing with photogrammetric imagery. Central perspective projection provides for a very powerful constraint, namely that of *epipolar geometry*, see figure 2.6. Given two images the so called *epipolar plane* for a point in 3D space (model or object space) is defined as the plane containing this point and the two projection centers of both images. This plane intersects both image planes in straight lines, the so called *epipolar lines*. If the relative

orientation of two images is known, for a given point in one image the epipolar line in the other image can be computed, and the corresponding point must lie on this epipolar lines. Thus the image matching problem is reduced from a two- to a one-dimensional task.

In order to facilitate matching along epipolar lines the two images can be transformed into the normal case in a preprocessing step, eliminating the vertical or y-parallaxes in the complete stereo model. Subsequently, matching only needs to be carried out along the (horizontal) direction of the base line. Note, that this preprocessing step is not required as such in order to take advantage of the epipolar constraint: for a given point in one image the epipolar line in the other image can be computed using the parameters of relative orientation, and matching can then be carried out along this epipolar line.

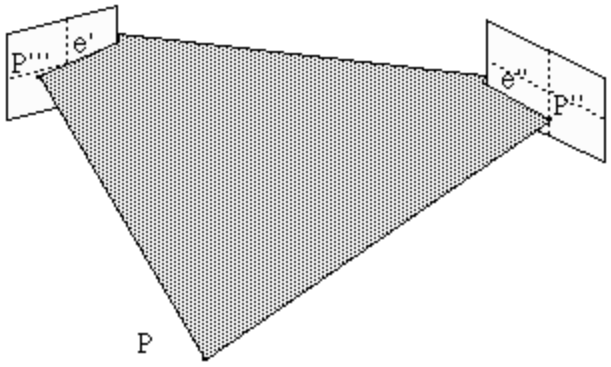

Figure 2.6: The epi-polar constraint: the epi-polar plane (P, P', P'') and the epi-polar lines e' and e''

The epipolar constraint is vital in reducing ambiguity problems and computational cost. Even if only approximate values for the parameters of relative orientation are known, the epipolar constraint should be used in order to restrict the search space for conjugate primitives in the direction perpendicular to the base line. Note that the epipolar constraint can only be formulated for pairs of images.

2.2.2.2.2 Object surface models

Geometric models for the object surface used in image matching range from horizontal and tilted planes to piecewise smooth surfaces, exhibiting discontinuities in the surface slope or the surface itself. Also, models borrowed from DTM generation such as finite element representations are used. As mentioned above the object surface is

assumed to be rigid, i.e. it does not change during the time interval between image acquisition.

Radiometric surface models describe the brightness of a pixel in object space, also called *groundel*. Due to deviations from the Lambertian (diffuse) reflection function, relief influences (shading), and other factors such as noise a groundel usually has a different brightness when viewed from different directions. In image matching these differences are usually modelled by means of a local linear radiometric transformation.

Thus, changes in overall brightness and contrast between different image patches are taken into account.

Another assumption of the object surface model is that it is opaque. This assumption guarantees that for a given primitive in one image there exists at most one corresponding primitive in each other image. In the case of occlusions, no corresponding primitive may exist.

2.2.2.2.3 Examples for sensor and object surface models in image matching algorithms

For cross correlation the two images are assumed to be of identical scale and azimuth, and to have parallel optical axes. In addition, the object surface is implicitly modeled as a local plane parallel to the image planes. This set up is equivalent to the so called *normal case* of photogrammetric image acquisition. In the simplest case the epipolar constraint is not used, however, it can be easily introduced by shifting the template matrix across the search matrix in a predefined direction only. The object surface is assumed to be opaque, and linear differences between the gray values of the two windows are allowed.

In least squares matching the rather strict geometric assumptions for cross correlation are relaxed: rather than only shifting the template matrix across the search matrix an affine transformation is used for the geometric mapping between the windows. As a result small deviations from the normal case can be tolerated, and the object surface is modeled as a local tilted plane. Again, the epipolar constraint can be easily introduced. The radiometric model is the same as for cross correlation. Figure 2.7 shows sensor and object surface model for object space least squares matching.

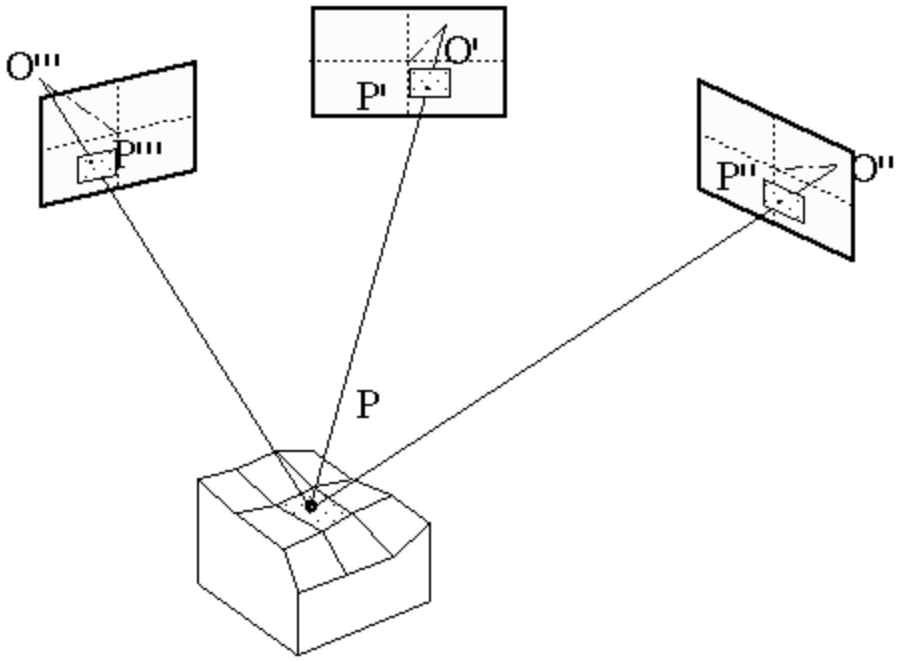

Figure 2.7: Sensor and object surface model for object space least squares matching

For ABM in object space the collinearity equations are explicitly set up. If local primitives are used, the object surface model is implicitly given by a tilted plane (Grün 1985). For global primitives a separate object surface model, often represented as connected bilinear surface patches, is introduced. This general model allows for the introduction of all orientation parameters (thus, the epipolar constraint is implicitly observed), and constraints for the geometric shape of the object surface such as parameters minimizing the surface curvature can be directly introduced. Within this model simultaneous multiple image matching can be carried out as discussed before. Again, the radiometric model is the same as for cross correlation.

In FBM sensor and object surface models are usually represented implicitly in order to reduce the search space. The epipolar constraint is used in most approaches. Due to the higher radiometric invariance of features as compared to gray value windows, radiometric models play a secondary role in FBM. They are, however, contained in the radiometric feature attributes.

2.2.2.3 SIMILARITY MEASURES AND OPTIMIZATION PROCEDURES

The definition of criteria for a good match obviously plays an important part in each matching algorithm. For ABM the similarity between gray value windows is defined as a function of the differences between the corresponding gray values. This function can be the covariance or the cross correlation coefficient between the windows, the sum of the absolute differences between corresponding pixels, or as is the case in least squares matching - the sum of the squares of the differences. These measures have their background in statistics and are theoretically well understood.

Defining a similarity measure for feature based matching is more complicated. The definition must be based on the attributes of the features. In most FBM algorithms the differences in the geometric and radiometric attribute values are combined using heuristics and thresholds in order to compute the similarity measure, called a cost function or benefit function. Whereas a cost function is to be minimized, a benefit function must be maximized in order to achieve a good match.

The optimization procedure which can be applied depends on the choice of the matching primitives. In local ABM an exhaustive search can be carried out as is the case in cross correlation. Alternatively, gradient based iterative schemes such as ordinary or robust least squares adjustment are available. The pull-in range for these approaches is rather small and lies in the range a few pixels only. Therefore, good initial values for the unknowns must be at hand. In order to subsequently achieve global consistency conjugate points are usually transformed into object space, e.g. via forward intersection. In this step the orientation parameters of the images may also be improved, leading to a bundle adjustment. The resulting 3D points are subsequently filtered, and blunders are detected and eliminated. In global ABM the optimization procedure, the generation of tree-dimensional information and the estimation of parameters describing the object surface are integrated into one model.

FBM starts with discrete features. Therefore, gradient based methods can not be employed for optimization. In local FBM for each given feature in one image a small search area is defined in the other image(s) using the selected mapping transformation. Subsequently, an exhaustive search is usually carried out in this search area. At this stage multiple matches may still be allowed. After all features of a certain region have been

processed, blunders are detected through global consistency checks similar to local ABM. Alternative schemes for global consistency shall only be mentioned here. They include relaxation labeling, simulated annealing, and dynamic programming. For relational matching tree search methods are employed.

2.2.3 THE MATCHING STRATEGY

An image matching algorithm consists of a number of steps. Each of the individual modules which can be employed for each step has advantages and disadvantages. Thus, potentially something is to be gained from suitably combining these modules. Moreover, some parameters such as the approximate overlap or an average terrain height must often be provided a priori in order to reduce the search space, and values for free parameters and thresholds (window sizes, criteria for stopping the optimization etc.) must be initialized. Finally, internal quality checks should be carried out in order to guarantee a correct result.

In the matching strategy the individual steps carried out within the algorithm are determined. This includes the input of prior information from a human operator, and the presentation of the results for final visual verification. In a comprehensive comparison between different images matching algorithms for photogrammetric applications Gülch (1994) showed that while under good condition accurate matching results can be achieved with a large variety of algorithms, a good matching strategy is decisive for a successful solution in more complicated situations. Faugeras et al. (1992) obtained a similar result for algorithms popular in computer vision. Some of the aspects of a good strategy are discussed in the following.

2.2.3.1 Hierarchy

Hierarchical methods are used in many matching algorithms in order to reduce the ambiguity problem and to extend the pull-in range. They are employed from coarse to fine, and results achieved on one resolution are considered as approximations for the next finer level. For this task images are represented in a variety of resolutions, leading to so called *image pyramids*. A typical image pyramid, in which the resolution from one level to the next is reduced by a factor of 2, is depicted in figure 2.8. A coarser resolution is equivalent to a smaller image scale, and a larger pixel size. Thus, the ratio between the

(fictitious) flying height and the terrain height increases as the resolution decreases, and local disturbances such as occlusions become less of a problem. Besides image pyramids, usually also a hierarchical representation of the object surface model is used.

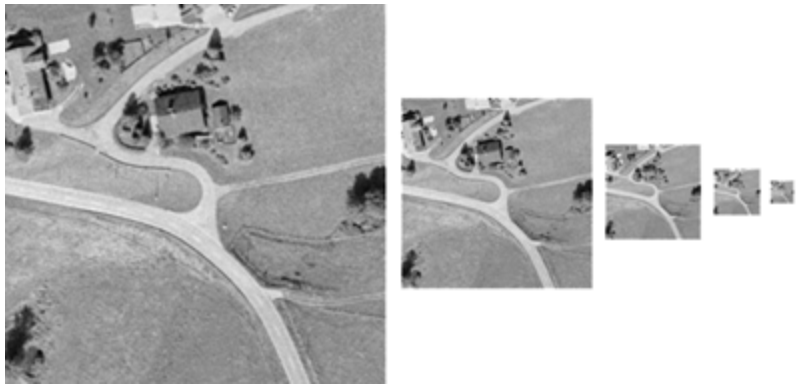

Figure 2.8: Example of an image pyramid

When FBM is used, feature extraction should be carried out on each resolution level separately, since features can vanish or be displaced from one level to the next due to the low pass filtering which is inherently present when decreasing the resolution.

2.2.3.2 Redundancy

It is not known how the human operator measures points stereoscopically, but he or she is certainly still more capable to set the measuring mark on the ground than any developed matching algorithm. In other words, the blunder rate for individually matched points can be rather high. Efficient blunder detection is only possible if there is a large redundancy in the system.

Therefore, it is prudent to determine many more points when using an automatic matching algorithm than a human operator would measure. It is also possible to do so, because the number of points to be measured is a secondary issue in an automatic procedure, as long as enough computational speed is available. A high point density can for instance be used to implicitly represent break-lines in a DTM. Also, single obstacle on top of a DTM such as houses or trees can be filtered out, if enough nearby points on the ground are given.

Another issue related to redundancy is that of multi image matching, and thus of object space matching. In a conventional photogrammetric block with 60 % end overlap and 20 % side overlap only 24 % of an image in the interior of the block is covered by two images, and the same area is covered by six images. Thus, multi image matching can be of advantage for DTM generation without having to acquire more images. Besides, it is a prerequisite for applications in aerial triangulation.

CHAPTER 3

METHODOLOGY

The major steps involved in methodology for completion of the objectives are geometric corrections, feature extraction, feature matching, mosaic creation and last step is the variation detection.

3.1. GEOMETRIC CORRECTIONS

The image sequences obtained from aerial images typically suffer from severe noise and brightness variations between corresponding images. In order to ensure uniform feature extraction from overlapping images for reliable matching the effect of these distortions must be minimized as much as possible.

To ensure uniform average brightness throughout the image set, it was decided to subtract the average brightness level of each image from its pixel data. This succeeds in bringing the image sequence into a reasonably uniform intensity level.

To reduce the effect of additive noise, a Gaussian smoothing function is applied to each image prior to feature extraction. The smoothing function eliminates any sharp regions of noise by performing a process similar to blurring.

Combined, the above two corrections were experimentally found to be reasonably effective in ensuring feature correspondence between overlapping images.

3.2. FEATURE EXTRACTION

Feature extraction is one of the most important first steps in Mosaicing. Its main objective is to find as many useful features from a scene while keeping the output noise level to a minimum. Edge, corner and vertex detection processes serve to simplify the analysis of images by drastically reducing the amount of data to be processed.

In order to perform a matching comparison between a set of images, there must be some reliable and reasonably stable criteria of correspondence. The most commonly employed feature points in feature point matching are corners and edges. These features are usually insensitive to noise and geometric distortions and have a very low probability of false positive matching.

There are a variety of algorithms for both corner and edge extraction. The most notable of these are canny edge detector, Harris corner detector and the SUSAN principle for extraction of both edges and corners.

The desired qualities of feature detectors are good detection (there should be a minimum number of false negatives and false positives), good localization (the edge location must be reported as close as possible to the correct position), response (there should be only one response to a single edge) and speed (the algorithm should be fast enough to be usable in the final Image Processing system).

For a real time system using time varying image sequences, speed is an important criterion to be considered, without too much compromise over the quality of results because spurious lines and edges can cause errors in motion analysis. Also there has to be a compromise between maximizing signal extraction and minimizing output noise.

Initial efforts towards feature extraction used small convolution masks such as Prewitt and Sobel operators to approximate the first derivative of the image brightness function, thus enhancing the edges. These filters give very little control on smoothing and edge localization.

The Canny edge detector was considered next, which has become one of the most widely used edge finding algorithms. The first step is defining and quantitatively developing criteria for edge detection (as given above) into a total error cost function. Variational calculus is applied to this cost function to find an optimal linear operator for convolution with the image. The optimal filter is shown to be a very close approximation to the first derivative of a Gaussian. Non maximum suppression in a direction perpendicular to the edge is applied to retain maxima in the image gradient. Finally the weak edges are removed using thresholding. The thresholding is applied using hysterisis.

The Gaussian convolution can be performed quickly because it is separable and a close approximation to it can be implemented recursively. However the hysterisis stage slows the overall algorithm considerably. The Canny edge detection method is found to be ten times slower than SUSAN approach. The results are stable FOR Canny but the edge connectivity at junction is poor and corners are rounded.

In the absence of multiple features, the SUSAN principle bypasses "the Uncertainty Principle of edge detection" which applies to most feature detectors (and most obviously the Gaussian based ones) with respect to Canny's first and second criteria.

Other methods include second order derivatives. The fact that SUSAN edge and corner enhancement uses no image derivative, explains why the performance in the presence of noise is good. The integrating effect of the principal together with its non-linear response gives strong noise rejection.

Keeping in view the anticipated operational requirements of the project, it was therefore decided to implement the SUSAN feature detector for both its speed and its ease of implementation.

3.2.1 THE SUSAN PRINCIPLE

The SUSAN principle is implemented using digital approximation of circular masks, (sometimes known as windows or kernels). If the brightness of each pixel within a mask is compared with the brightness of that mask's nucleus, then an area of the mask can be defined which has the same (or similar) brightness as the nucleus.

The concept of each image point having associated with it a local area of similar brightness is the basis for the SUSAN principle. This area is known as USAN (Univalue Segment Assimilating Nucleus) and contains much information about the structure of the image. From the size, centroid and second moment of the USAN, two dimensional features and edges can be detected. No image derivatives are used and no noise reduction is needed.

As seen in the figures 3.1 and 3.2, the USAN area is at a maximum when the nucleus lies in a flat region of the image surface. It falls to half of this maximum very near a straight edge and falls even further when inside a corner. This property of USAN's area is used as the main determinant of the presence of the edges and two-dimensional features.

The working of the SUSAN principle is explained in the diagrams shown below.

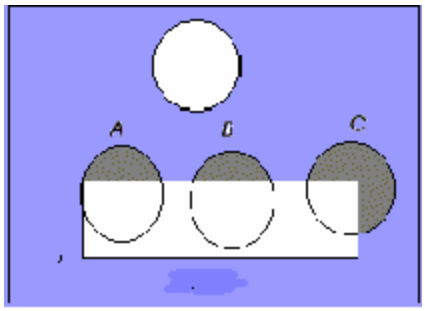

Figure 3.1. Four circular masks at different places on a simple image.

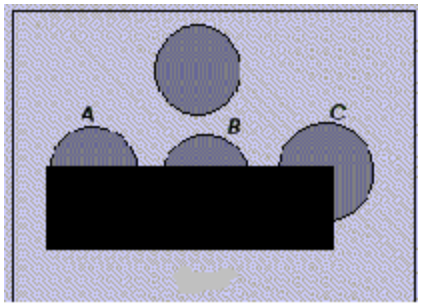

Figure 3.2. Four circular masks with similarity coloring; inverted USANs are shown as grey parts of the masks.

Consideration of the above arguments and observation of examples lead directly to the formulation of the SUSAN principle. An image processed to give as output inverted USAN area has edges and two-dimensional features strongly enhanced with the two-dimensional features more strongly enhanced than edges. This gives rise to the acronym **SUSAN (Smallest Univalue Segment Assimilating Nucleus).** The strength of the SUSAN principle is that the use of controlling parameters is much simpler and less arbitrary and therefore easier to automate than other edge detection algorithms.

3.2.1.1 SUSAN Edge Detector

The following steps are performed at each image pixel. Place a circular mask around the pixel in question. Calculate the number of pixels within the circular mask which have similar brightness to the nucleus. These define the USAN. Subtract USAN size from geometric threshold to produce edge strength image. Use moment calculations applied to the USAN to find edge direction. Apply non-maximum suppression thinning and sub-pixel estimation, if required.

SUSAN edge finder has been implemented using digitally approximate circular masks giving a mask of 37 pixels or a three by three mask. The masks are used either with constant weighting within them or with Gaussian weighting. The mask is placed at each point in the image and, for each point; the brightness of each pixel within the mask is compared with that of the nucleus (the center point).

If the difference of the brightness values is less than a threshold "t", then the output of the comparison is 1 else its 0. The comparison is done for each pixel within the mask and is summed up. This total "N" is just the number of pixels in the USAN, i.e., it gives the USAN's area. The parameter "t" determines the maximum contrast of features which will be detected and also the minimum amount of noise which will be ignored.

Next, N is compared with a fixed threshold "g", which is set at 3*Nmax/4. This value is calculated from analysis of the expectation value of the response in the presence of noise only. The use of 'g' should not result in incorrect dismissal of correct edges.

The algorithm described gives good results but a more stable and sensible equation is used for 'C' in place of equation 1.

$$C = \exp((d/t)*6)$$

This gives a smoother version of equation 1 as it allows pixel brightness to vary slightly without having too large an effect on 'C', even if it is near the threshold position. This form gives a balance between good stability about the threshold and the function originally required.

3.2.1.2 Computation of Edge direction

Computation of edge direction is needed for a variety of reasons. If non-maximal suppression is to be performed, the edge direction must be found. It is also necessary if edges are to be localized to sub-pixel accuracy. Finally, applications using the final edges often use the edge direction for each edge point as well as its position and strength. In case of most edge detectors, edge direction is found as part of the edge enhancement. As SUSAN principle does not require edge direction to be found for enhancement to take place, a reliable method of finding it from USAN has been developed.

The direction of an edge associated with an image is found by in two ways depending on the type of edge points, namely inter-pixel edge case, and intra-pixel edge case. The edge direction is calculated by finding the longest axis of symmetry of USAN.

This is found by calculating the following sums:

$$m(X - Xo)^2 * (Ro) = SUM[((X - Xo)^2 * C(R,Ro)] \quad ---[1]$$

$$m(Y - Yo)^2 * (Ro) = SUM[((Y - Yo)^2 * C(R,Ro)] \quad ---[2]$$

$$m(X - Xo)(Y - Yo)*(Ro) = SUM[(((X - Xo)(Y - Yo)*C(R,Ro))] \quad ---[3]$$

Where,

Ro: Nucleus

R (Ro): Center of gravity (of USAN of Ro)

C = e (d/t)*6

m: mean

SUM: Summation

Ratio of equations [1] and [2] gives the orientation of the edge. Equation [3] gives the sign of the gradient.

We find the sum of second moments of USAN about the nucleus to find the orientation of the edge. This can be found to varying accuracy depending on the mask used.

The edge response obtained from above is suppressed so that non-maxima (in the direction perpendicular to the edge) are prevented from being reported as edge points).

Following this, the “strength thinned” image can be binary thinned using standard thinning processes.

3.2.1.3 SUSAN Corner Detector

This is very similar to the edge detector, particularly in the earlier stages. The steps required for finding the corners are as follows.

Place a circular mask around the pixel in question. Calculate the number of pixels within the circular mask which have similar brightness to the nucleus. These define the USAN. Subtract USAN size from geometric threshold (which is set lower for corners) to produce a corner strength image. Test for false positives by finding the USAN's centroid and its contiguity. Use non-maximal suppression to find the corners.

All pixels within a circular mask are compared and the response is summed up in exactly the same way as in edge detector. Again, here the sum is compared with a geometric threshold "g". Here, the corner detector is quite different from the edge detector, where "g" was necessary only in the presence of noise. For a corner to be present, the sum must be less than half the maximum possible sum. Therefore, "g" is set to exactly half of maximum sum.

Sometimes, SUSAN will give false positives in certain circumstances. This can occur with real data where blurring of boundaries between regions occurs and there is a thin line half way between the two surrounding regions. It may cause corners to be wrongly reported.

The problem has been eliminated by the following method. The center of gravity of USAN and its distance of this from the nucleus is found. A proper corner will have center of gravity that is not near the nucleus and thus false corners can be rejected.

A final addition to SUSAN principle is a simple rule, which enforces "Contiguity" in USAN. This is necessary for images having lot of noise and fine complicated structure. All pixels within a mask, lying in a straight line pointing outwards from the nucleus in the direction of center of gravity of USAN must be a part of the USAN, for a corner to be detected. This is effective in forcing the USAN to have a degree of uniformity, and reduces false positives.

The corners detected by the implementation of the SUSAN algorithm are as shown in the figure 3.3.

Figure 3.3 Corner Detection

3.3 FEATURE MATCHING

Once an appropriate number of corners have been extracted in each of the component images of the input set, the major part of the system can then be invoked.

Despite the availability of a large number of very complex algorithms for feature correspondence, it was decided to design an original algorithm which would be reasonably fast to fit into the overall performance requirements of the system.

It was decided to carry out the feature point matching in two steps. The first step is responsible for finding out a tentative area of overlap between a pair of images. Once a tentative area of overlap has been calculated, this overlap is then confirmed using corner matching in the overlapping area. If both these steps indicate a match, then the image pair is guaranteed to be neighbors.

The feature correspondence algorithm implemented in the system can be depicted as in figure 3.4. Here, a number of images are processed and as an end product, one get a single mosaic of many images and the mosaic is free from any overlapping.

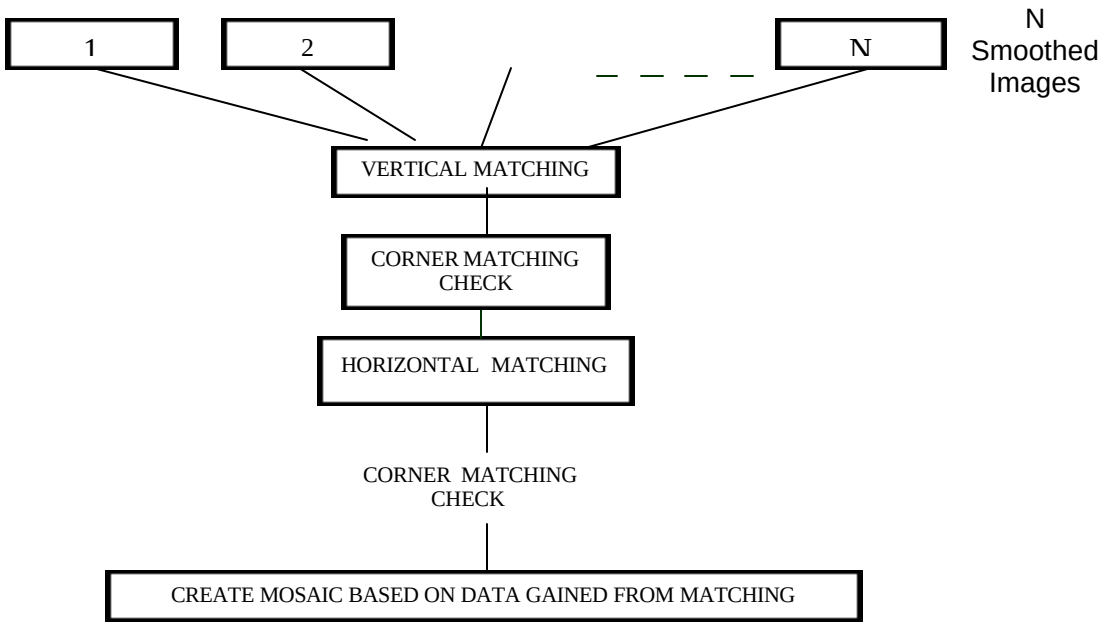

Figure 3.4 The feature Correspondence Algorithm

The matching algorithm is responsible for creating a mapping for each image in the image sequence, determining its position in the final mosaic.

3.3.1 DERIVATIVE MATCHING

To calculate tentative overlaps, it was decided to use image derivative matching for many reasons because image derivates are extremely fast to calculate, image derivates are reasonably immune to global changes like shading etc., image derivates give a much greater appreciation of sharp changes (features) in the image, there is very little probability of non matching images to have same derivates at same locations (suppression of false positives).

To calculate image derivates, the following equations were used: -

$$f'(x,y) = f(x+1,y) - f(x,y) \qquad \text{(for x-derivative)}$$

$$f'(x,y) = f(x,y+1) - f(x,y) \qquad \text{(for y-derivate)}$$

The implementation of the above equations when executed on the images highlight areas of change (features) and are thus much more suitable for matching than the original images.

The derivative matching algorithm then proceeds as is explained below. The procedure is that an image is picked at random and is checked for vertical overlap with all other images in the image set by employing image derivate strips for comparison. This step is also repeated fro horizontal overlap. If there is any match in horizontal or vertical, this information is stored in the mapping data structure for later use in mosaicing. The same steps are repeated for all remaining images.

At the completion of the algorithm's execution, the program has enough information to proceed with the confirmation of the tentative overlap using corner matching.

3.3.2 ERRORS AND THRESHOLDS

When comparing pixel derivatives, there are two types of errors. The first is the local error in comparison between single pixel derivative values. And the other is the global error which is the number of pixel derivatives that fail to match.

There are self adjusting thresholds for both types of errors. Thresholds adjust on the basis of brightness and contrast in the local area. This adjusting of thresholds has a large effect on the eventual matching of images. Figure 3.5 displays the step-wise procedure for finding tentative boundary.

Figure 3.5 Finding Tentative Boundary

3.3.3 CORNER MATCHING

The aim of the corner matching phase is to confirm the existence of the overlap calculated in the derivate matching phase. Although image derivatives provide sufficient results for images with low complexity, they start to fail on images with considerable

detail. For this, the results of derivative matching must be confirmed using a more dependable technique. Corner matching was used because corners are usually faster to extract than other features and also the matching of corners is relatively straightforward.

The corner matching algorithm proceeds as explained below. After establishment of tentative boundaries, the program must correlate corners between two images. The system uses **Sum of Squared Differences** method to correlate corners. A correlation window in the overlap area is selected, as shown in figure 3.6. The strengths of corners in the correlation window are compared. These differences are squared and added. The correlation window is moved throughout the overlap area.

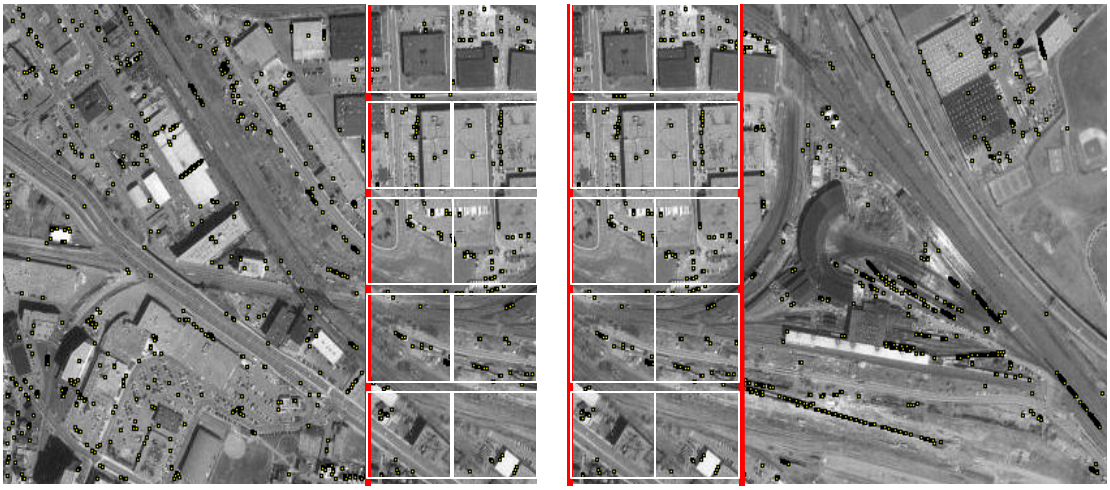

Figure 3.6 Correlations Windows

3.4 CREATION OF THE MOSAIC

The final step in the mosaicing sequence is the creation of the mosaic itself. At the end of the corner extraction and feature matching phases, one has information about the position of each image component in the final mosaic and its overlapping area with immediate neighbors (top, bottom, left, and right). This information is then used in the mosaic creation phase. This phase involves the following tasks.

First, allocation of image size based on calculations of component image sizes and overlap regions is made. Secondly, copying of component images onto corresponding locations on the mosaic is carried out. Lastly, removal of sharp joins using blending is accomplished.

The result is the creation of a single image that is seamlessly blended and no overlapping of any type can be observed.

The image can be saved in the permanent memory as well for subsequent change in variation detection.

3.5 VARIATION DETECTION

The last step is to compare two images, of the same area, for changes, if any. This is accomplished by taking image derivatives at each point in the two images, and if a difference greater than a particular threshold is found, that area is identified in the form of red rectangles.

- Input the two mosaics to be compared for differences
- Create a rectangular mask sized according to input parameters.

While (image limits not exceeded)

Place the mask at the same location in both mosaics

Take image derivates for all image pixels under the mask.

For (all pixels under the mask)

Take derivates of both images

Compare the derivates.

Update error count if they differ by more than thresh-hold.

If (error count > masksize/2) mark the mask boundaries as change boundaries.

Advance mask by masksize/2.

end while.

End.

CHAPTER 4

IMPLEMENTATION DETAILS

4.1 ARCHITECTURAL DESIGN

The architecture of the software is as shown below: -

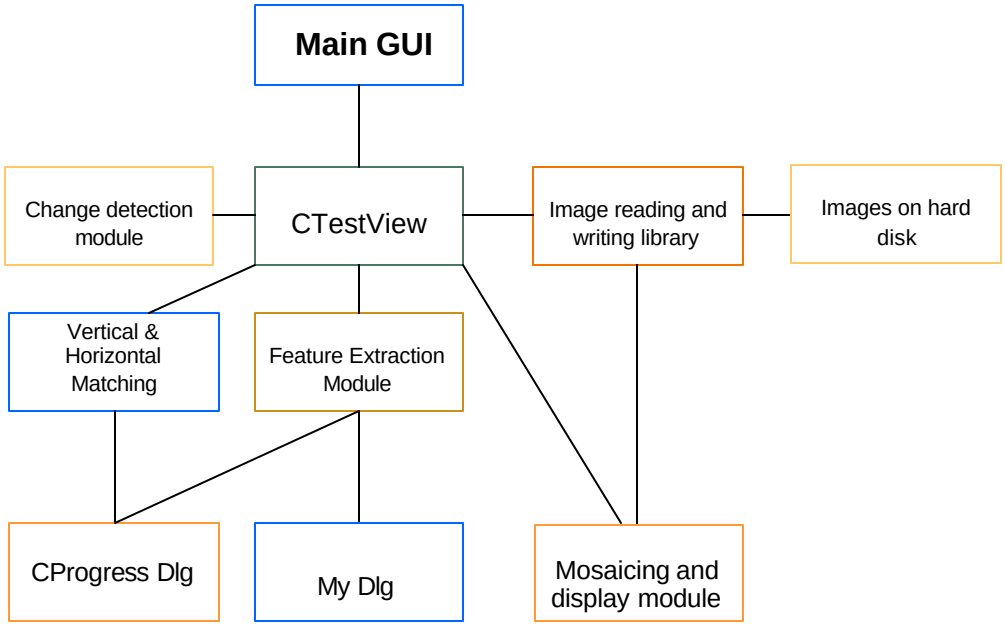

Fig. 4.1 The Architecture of the Software

The main GUI interacts with the CTestView class. Once the user request is received by the CVieClass, it calls all other modules and uses underlying libraries to respond to the user request.

The project was developed using Microsoft's VISION SDK (Software Development Kit) library as an implementation base.

4.2 VISION SDK

The project was developed using Microsoft's VISION SDK (Software Development Kit) library as an implementation base.

The Microsoft Vision SDK is a library for writing programs to perform image manipulation and analysis on computers running Microsoft Windows operating systems.

The Microsoft Vision SDK was developed by the Vision Technology Research Group in Microsoft Research to support researchers and developers of advanced applications, including real-time image-processing applications. It is a low-level library, intended to provide a strong programming foundation for research and application development; it is not a high-level platform for end-users to experiment with imaging operations. The Microsoft Vision SDK includes classes and functions for working with images, but it does not include image-processing functions. The Microsoft Vision SDK is a C++ library of object definitions, related software, and documentation for use with Microsoft Visual C++.

Compared to other typical packages for image processing, it has four key virtues and two flaws: The virtues are that it is suitable for fast, real-time image processing, it has a nice interface to Windows, such as shared image memory across processes and GDI interface, it has user-definable pixel types (very important for research) and that it has a device-independent interface for image acquisition. It can be used to create binaries that can be transported and run on a machine with any supported digitizer or camera. It can be easily extended to support new types of digitizers or cameras.

The flaws are that the Vision SDK assumes that images are resident entirely in RAM. There is no general support for very large image files that cannot be brought into RAM in their entirety. The other flaw is that The Vision SDK does not include image-processing operators. The Vision SDK is best thought of as a low-level substrate for developing computer vision and/or image processing programs or systems, giving a nice interface to the operating system but not providing high-level image-processing operators.

4.3 HEADER FILE DETAILS

The following header files are included in the project:

4.3.1 DataArray.h

This file includes the data structure for storing the information about the neighborhood of each image. Pointers for up, down, left and right images are included. In addition overlaps and matching errors for the images are also stored. The up, down, left and right pointers are initialized to -1. As the matching takes place they are set to the number of whatever image is present at the respective locations.

4.3.2 MainFrm.h

It includes the class description for the CMainFrame class derived from the CFrameWnd class. It contains declarations for the class which represents the frame window.

4.3.3 MyDlg.h

It contains declarations for the class which represents an input dialog box. This dialog box inputs parameters for the corner detection phase.

4.3.4 ProgressDlg.h

It contains declarations for the class representing the dialog box which contains a progress bar. This progress bar is used to show progress of the corner detection phase and the image mosaicing phase.

4.3.5 StdAfx.h

Contains declaration for the windows code.

4.3.6 Test.h

Contains class declarations for the CTestApp class which represents the windows application.

4.3.7 Testdoc.h

It contains class declarations for the CTestDoc document class.

4.3. 8 TestView.h

It contains the declarations for the CTestView class which is the main class of the project and contains declarations for all the variables of the technical part of the project.

4.4 CODE FILE DETAILS

The following code files are included in the project:

4.4.1 MainFrm.cpp

This file contains the implementation of the CMainFrame class which is derived from the CFrameWnd class and represents the frame window.

4.4.2 MyDlg.cpp

This file contains the implementation of the MyDlg class which takes in the input parameters for Corner Detection. It has four member variables

m_masksize, m_threshold, m_usan, m_usanl which contain values of the masksize , threshold , upper USAN area and lower USAN area respectively once the dialog box is used.

4.4.3 ProgressDlg.cpp

This file contains the implementation of the CProgressDlg class which is responsible for handling the dialog containing the progress bar.

4.4.4 StdAfx.cpp

It is the source file contains the standard includes for a windows program.

4.4.5 Test.cpp

This file contains the implementation details of the CTestApp class which represents the main windows application.

4.4.6 Testdoc.cpp

It contains implementation of the CTestDoc class which represents the document related to the View in this program.

4.4.7 TestView.cpp

It contains the definitions of all the major functions of the system and is the main file of the program. All event handling and image processing tasks are handled with this file.

A list of the major functions in the main file of the project is given below, followed by a brief explanation of each of the major methods: -

4.5 IMAGE PROCESSING FUNCTIONS

The following are the main image processing functions developed are explained below.

4.5.1 The SmoothImage ( ) function

The SmoothImage ( ) function is defined in the TestView.cpp file and it is the first major image processing function to be called for each image.

The input to this function is a raw image from the image set in the form of a CVisRGBAByteImage Object. The purpose of this function is to remove any random

noise in the image as well as remove insignificant details to make the image more suitable for the matching phase.

This function performs the bulk of the geometric correction portion of the project. It accomplishes this by defining a kernel and then performing Gaussian smoothing of the image by employing this kernel. This is achieved by changing the grey levels of the image at each pixel by ratios determined by the kernel coefficients , such that each image position better reflects the overall gray level in its neighborhood.The function is called for each image in the image set to be mosaiced.

4.5.2 The Detect-Corners( ) function

The Detect_Corners function is defined in the TestView.cpp file. This function forms the backbone of the feature extraction part of the system. The function takes as input a single smoothed image in the form of a CVisRGBAByteImage object and performs corner extraction on it. The detected corners, their locations and corner strengths are stored in a data structure for use in the matching phase.

This function is an implementation of the SUSAN principle for corner detection. The function detects features (corners in this case) based upon certain control parameters input by the user at run time. These parameters control the amount of detail to be explored in feature extraction, as well as the number of features extracted.

4.5.3 The VerticalMatch ( ) function

The VerticalMatch function is defined in the TestView.cpp file. This function is responsible for the matching of extracted features in pairs of images in conjunction with the HorizontalMatch function.

The inputs to this function are two smoothed images in the form of CVisImageRGBAByteImage objects. The function carries out vertical feature matching on the pair of images. If there is a match, the function also calculates the amount of overlap between the images. It then stores the neighbor and overlap information in a data structure for use in the mosaic creation phase.

For vertical overlap detection, the algorithm assigns one image in the pair as the reference image and performs a vertical scan of the other image for an exact match of extracted features.

The function first carries out a derivative matching of image strips to calculate a tentative matching area in the pair of images. This tentative overlap is then confirmed using corner matching. If both these criteria are satisfied, the images are assumed to have matched and the information is stored. The process is repeated for all images in the set.

4.5.4 The HorizontalMatch( ) function

The HorizontalMatch function is defined in the TestView.cpp file . This function is responsible for the matching of extracted features in pairs of images in conjunction with the VerticalMatch function.

The inputs to this function are two smoothed images in the form of CVisImageRGBAByteImage objects. The function carries out horizontal feature matching on the pair of images. If there is a match, the function also calculates the amount of overlap between the images. It then stores the neighbor and overlap information in a data structure for use in the mosaic creation phase.

For overlap detection, the algorithm assigns one image in the pair as the reference image and performs a vertical scan of the other image for an exact match of extracted features.

The function first carries out a derivative matching of image strips to calculate a tentative matching area in the pair of images. This tentative overlap is then confirmed using corner matching. If both these criteria are satisfied, the images are assumed to have matched and the information is stored. The process is repeated for all images in the set.

4.5.5 The Mosaic( ) function

This function is defined in the TestView.cpp file. The function is the last major function to be called in the image mosaicing sequence. The function operates on the entire set of images initially loaded into the program for the purpose of creating a single mosaic.

Once the matching phase is complete and the program has enough information about the position of each component in the final mosaic, this function is called to carry out the

actual process of building the mosaic. Based on information obtained in the matching phase the overlapping areas are removed with extreme accuracy. The function returns with the final mosaic as the result.

The function operates by first of all sorting the data structure obtained as a result of the feature matching phase. The purpose of sorting the data structure is two fold. Firstly, the sorted data structure exactly depicts the position of each image component in the final mosaic. All that remains then is to copy each component from the data structure into its correct position. Secondly, sorting the data structure makes it possible to calculate the final size of the mosaic of the mosaic that has to be allotted in a more efficient manner. This involves removing the overlapping areas which can only be determined once the image components have been sorted.

Once the sorting and image copying is complete, the final mosaic is then displayed in the main window of the program.

4.6 INTERFACE CONTROL FUNCTIONS

The following functions have been used for Interface control.

4.6.1 The ReadImages( ) function

This function is defined in the TestView.cpp file and is responsible for reading in the selected image set from the disk into RAM and converting them into objects of the CVisRGBAByteImage class for all subsequent manipulations in the program.

4.6.2 afx_msg void OnDisplay( ) function

This is a message handler for the Display menu item in the program's main menu bar. The function is defined in the TestView.cpp file and is responsible for displaying the selected image in a separate window.

4.6.3 afx_msg void OnUpdateClear( ) function

This is a message handler for the Clear menu item in the main menu bar of the program . The function is defined in the TestVeiw.cpp file and is responsible for clearing the RAM of all currently loaded images. This is achieved by destroying all CVisRGBAByteImage objects . This relieves the RAM after each execution of the program so more images can be loaded for a new execution.

4.6.4 afx_msg void OnCorners( )function

This is a message handler for the CornerDetection menu item in the main menu bar of the program. The method is defined in the TestView.cpp class. This function performs some cleaning up of the corner storage data structure and then calls the detect_corners function for subsequent feature extraction.

The above discussion gives a list and detail of only the more important and relevant functions in the project. Apart from these, a lot of small helper functions were created in the major files which were concerned with the details of Visual C++ programming structure.

CHAPTER 5

RESULTS AND ANALYSIS

5.1 RESULTS

The horizontal mosaicing of images is shown in Figure 5-1. Two vertically photographed aerial images of a scene are taken and then mosaiced into a single image at the bottom. Notice the seamless blending at the edges and virtually no loss or duplication of information.

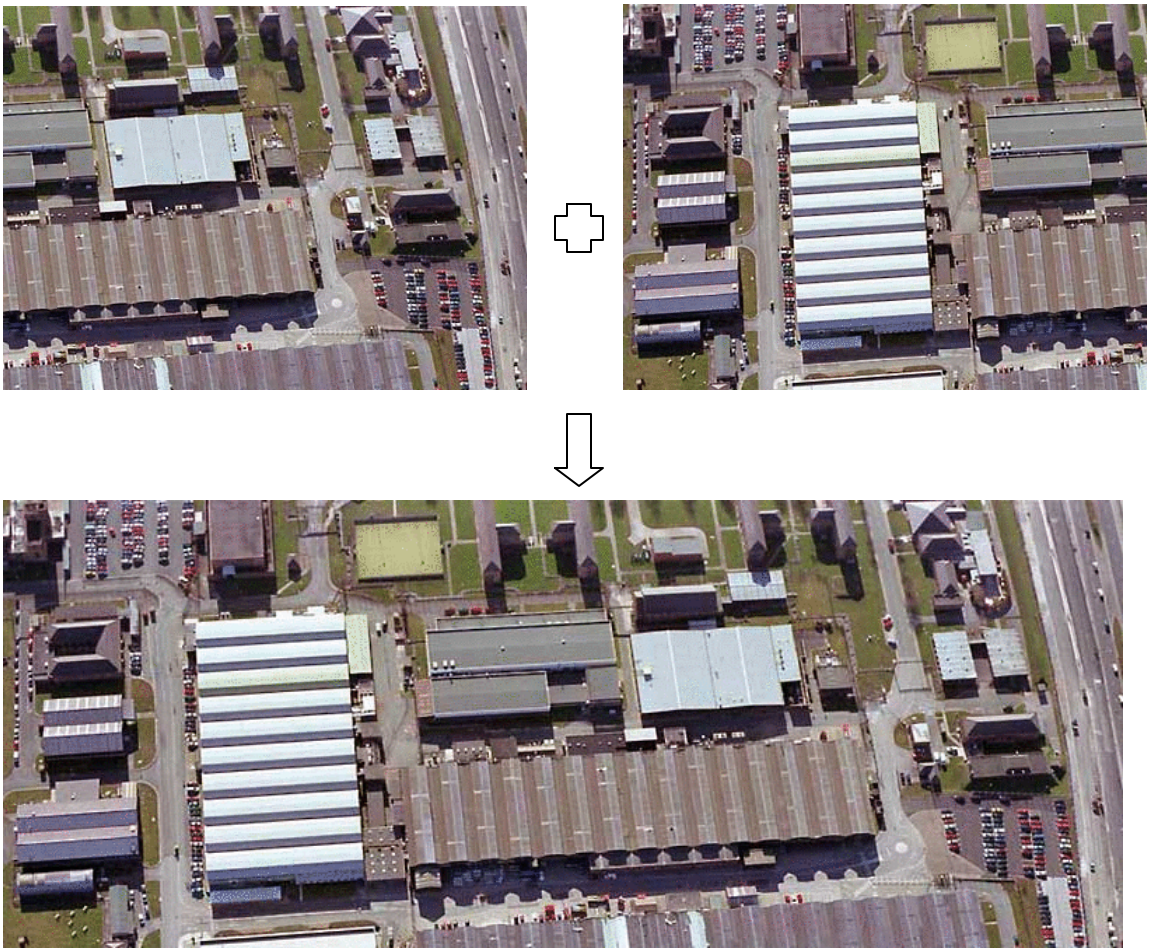

Figure 5.1 Horizontal Mosaicing of two images

An example of the use of the program to mosaic more than two images is illustrated in figure 5.2. Six images which match both vertically and horizontally are mosaiced into a single panorama. The blending problem can be removed by including an appropriate blending filter in the program.

Figure 5.2 Mosaicing of six images

A very good example of how the program is detecting corners can be seen in figure 5.3. The number of corners has been suppressed by a mask and filter. If required it can be further suppressed or relaxed as required by the program.

Figure 5.3 Example of corner detection

Change detection as performed by the program is illustrated in Figure 5.4. The top image is the changed image and the program has indicated the appropriate changes in the original image at the bottom. The use of small boxes is used to approximate the shape of the change object introduced. The car in the top image has been approximated with small boxes in the bottom image.

Figure 5.4 Variation Detection in Images

5.2 ANALYSIS

When compared with other techniques, the following results were observed. Our implementation of the not only outperformed other algorithms in speed but also in consistency. The speed was mainly due to use of feature extraction instead of pixel by pixel matching and consistency was due to extra checks incorporated to prevent any false positives.

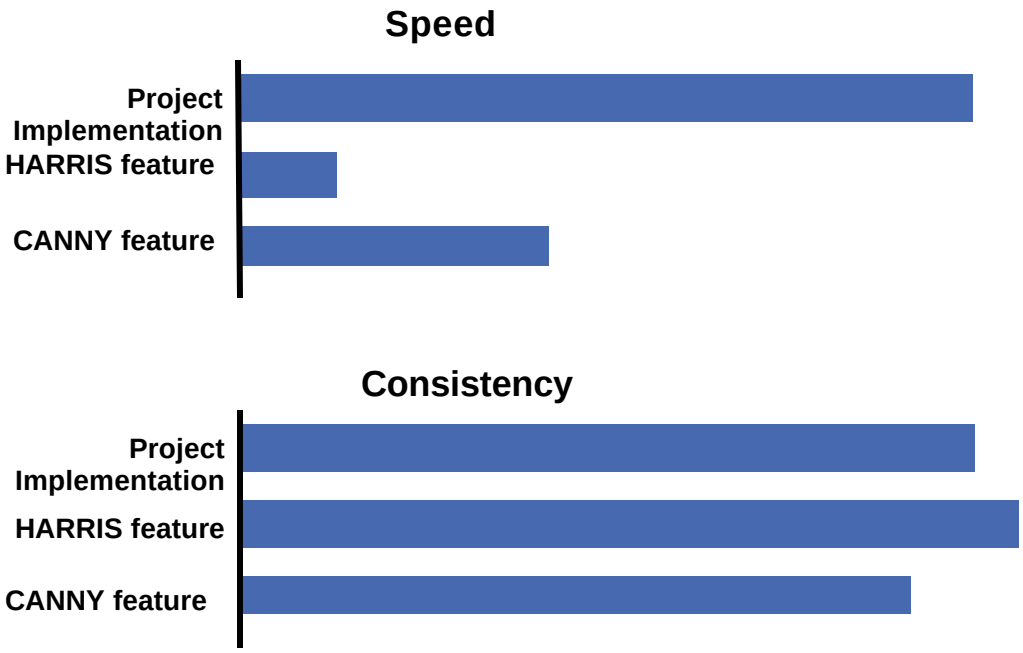

Figure 5.5 Speed and Consistency

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

In conclusion one must look at the objectives that were defined at the beginning of the project and evaluate whether they have been achieved at the end. The primary objective was to develop a system to achieve the seamless mosaicing of a number of images into a composite whole. Considering the operation of our program on some test images it can be safely said that objective has been achieved to a large extent. In all of the test images that were run on this program, it provided the correct results more than 90% of the time. Furthermore it was able to mosaic up to 16 images of an area which is a good indicator of its robustness.

A secondary objective was to develop an efficient technique for the matching of images. The technique of using feature extraction and correlation to match two images and keep error counts has been more than successful in this regard. The program has been able to create mosaics of 16 images of 800x600 resolution in around 4 minutes which is many orders of magnitude better than conventional image matching techniques. Fourier Transforms and Phase Correlation which are normally provide equal results but are not as efficient.

The third objective of detecting changes between two images of the same area was also completed amenably. Change Detection technique employed here compares images on the basis of not only the change of region but also of the features extracted earlier. This allows the program to approximate to a large extent the actual shape of the object which has been introduced as a change into the new image. In addition the program has displayed good ability to reject minor changes introduced because of noise and distortion.

Although introducing noise robustness in this program was not within the scope, this was achieved as an added objective. The software can tolerate a fair degree of noise – which may include oil blurring, Gaussian noise and simple distortion. It does not remove the noise, instead it correctly determines if the noisy image joins with another image. This capability is due to the efficient and robust image matching technique used.

One of the shortcomings of the software it that it does not tolerate angle of view shifts between two images to be mosaiced. If both have significant amount of shift between the angles of view of the camera then results might not be that good. This problem can be rectified by dovetailing this program with software that performs affine transformation between different images. Once the transformations have been performed the software can work on any type of images.

6.2 FUTURE WORK

A lot of capabilities can be added to the basic design of the software. In the case of image mosaicing, a camera distortion correction module can be an effective addition. Also, the program can be made capable to mosaic images which are not rectangular in shape. This can be achieved with some loss in efficiency of the program but it will enhance to a large extent the range of applications that this program will be able to perform.

In addition, the very effective change detection algorithm enables the approximation of the shape of the new object introduced in the picture. If a matching algorithm is developed which can match a detected shape with templates of some common shapes such as cars, buildings, trees, water reservoirs etc stored in a library then excellent capabilities of object recognition can also be inculcated in the program without much effort.

To conclude, solid groundwork has been laid in the designing of this software. Not only all objectives defined at the beginning have been achieved satisfactorily, but also the design provides excellent opportunities for anyone interested in enhancing the capabilities of this software.

BIBLIOGRAPHY

1. Ackermann F., 1995: Automatic Aerotriangulation, Proceedings, 2nd Course in Digital Photogrammetry, Landesvermessungsamt Nordrhein-Westfalen und Institut für Photogrammetrie, Universität Bonn.
2. Baltsavias E., 1991: Multiphoto geometrically constrained matching, Dissertation, Institut für Geodäsie und Photogrammetrie, Mitteilungen der ETH Zürich (49).
3. Barnard S.T., 1987: Stereo matching by hierarchical microcanonical annealing, SRI International, Technical note 414.
4. Barnard S.T., Thompson W.B., 1980: Disparity analysis of images, IEEE-PAMI (2) 4, 333-340.
5. Cho W., 1995: Relational matching for automatic orientation, PhD thesis, Department of Geodetic Science and Surveying, The Ohio State University, Columbus, OH.
6. Ebner H., Hoffmann-Wellenhof B., Reiß P., Steidler F., 1980: HIFI - A minicomputer program package for height interpolation by finite elements, IntArchPhRS (23) B4, 202-215.
7. Ebner H., Fritsch D., Gillessen W., Heipke C., 1987: Integration von Bildzuordnung und Objektrekonstruktion innerhalb der digitalen Photogrammetrie, BuL (55) 5, 194-203.
8. Faugeras O., Fua P., Hotz B., Ma R., Robert L., Thonnat M., Zhang Z., 1992: Quantitative and qualitative comparison of some area and feature-based stereo algorithms, in: Förstner W., Ruhwiedel S. (Eds.), Robust Computer Vision, Wichmann, Karlsruhe, 1-26.
9. Förstner W., 1982: On the geometric precision of digital correlation, IntArchPhRS (24) 3, 176-189.
10. Förstner W., 1986: A feature based correspondence algorithm for image matching, IntArchPhRS (26) 3/3, 150-166.
11. Gülch E., 1994: Erzeugung digitaler Geländemodelle durch automatische Bildzuordnung, DGK-C 418.

12. Grün A., 1985: Adaptive least squares correlation: a powerful image matching technique, South African Journal of Photogrammetry, Remote Sensing and Cartography (14) 3, 175-187.
13. Hannah M.J., 1989: A system for digital stereo image matching, PE&RS (55) 12, 1765-1770.
14. Heipke C., 1990: Integration von digitaler Bildzuordnung, Punktbestimmung, Oberflächenrekonstruktion und Orthoprojektion in der digitalen Photogrammetrie, DGK-C 366.
15. Heipke C. 1995: Digitale photogrammetrische Arbeitsstationen, DGK-C 1995.
16. Helava U.V., 1988: Object-space least-squares correlation, PE&RS (54) 6, 711-714.
17. Hobrough G.L., 1959: Automatic stereo plotting, PE&RS (25) 5, 763-769.
18. Kölbl O., Bach U., Gaisor D., de Laporte K., 1992: Multi-template-matching for the automation of photogrammetric measurements, IntArchPhRS (29) B3, 540-548.
19. Kreiling W., 1976: Automatische Erstellung von Höhenmodellen und Orthophotos durch digitale Korrelation, Dissertation, Institut für Photogrammetrie, Universität Karlsruhe.
20. Mayr W., 1995: Aspects of automatic aerotriangulation, in: Fritsch D., Hobbie D. (Eds.), Photogrammetric Week '95, Wichmann, Karlsruhe, 225-234.